Tentamen Algoritmiek
Donderdag 9 juli 2015, 10:00 – 13:00 uur

Geef een **duidelijke toelichting** bij al je antwoorden.
Veel succes !

Opgave 1. (20 punten)
We bekijken een tweepersoonsspel genaamd *Cram*, dat meestal gespeeld wordt op een rechthoekig bord van m bij n, met m het aantal rijen en n het aantal kolommen ($m, n \geq 1$). Bij aanvang van het spel is het bord leeg. Twee spelers, in deze opgave Josephine en Napoleon, leggen om de beurt een dominosteen op het bord. Een domino kan horizontaal of verticaal worden neergelegd, en bedekt altijd precies twee naburige vakjes. Het spel is afgelopen zodra een der spelers geen dominosteen meer kan plaatsen. In dat geval heeft de tegenstander gewonnen; die heeft de laatste zet gedaan. We spreken af dat Josephine begint.

Voorbeeld van een mogelijk spelverloop voor $m = n = 4$ vanuit de beginsituatie. Lege vakjes geven we aan met `-`, door een domino bezette vakjes met een `X`. Een domino ziet er dus uit als `X X` (horizontaal neergelegd) of $\begin{matrix}\texttt{X}\\\texttt{X}\end{matrix}$ (verticaal neergelegd).

```
- - - -          - - - -          - - - -          X X - -          X X - -
- - - -  --->    - X X -  --->    - X X -  --->    - X X -  --->    - X X -
- - - -   J      - - - -   N      - - - X   J      - - - X   N      X X - X
- - - -          - - - -          - - - X          - - - X          - - - X
                                                                      (&)
```

In toestand `(&)` is Josephine aan de beurt.

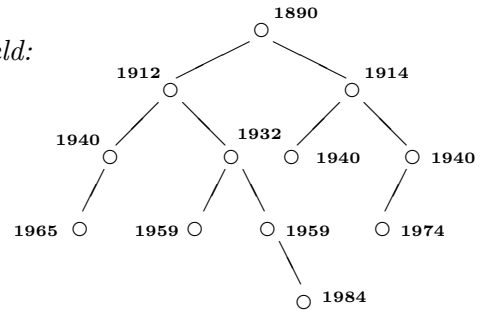
a. (2 punten)
Wat zijn voor dit spel toestanden en acties (voor algemene m en n)?
b. (12 punten)
Teken de toestand-actie-ruimte voor het geval $m = n = 4$, uitgaande van toestand `(&)`, waar Josephine aan de beurt is. Geef bij *elke* toestand in je toestand-actie-ruimte aan of deze winnend is voor J of voor N, te beginnen bij de eindstanden. Bepaal zo of `(&)` winnend is voor J of N en leg uit hoe gespeeld moet worden om te winnen.
Toestanden waarvan de laatste zetten vastliggen hoef je niet verder uit te werken. Je kunt daar meteen bijzetten wie er wint. Tip: draai je papier een kwart slag (landscape) om genoeg ruimte te hebben.
c. (6 punten)
We bekijken het geval dat $m = 2$ (2 rijen dus). Duidelijk is dat het spel voor $m = 2, n = 1$ winnend is voor de beginnende speler, en $m = 2, n = 2$ juist winnend voor diens tegenstander. Laat zien, zonder de volledige toestand-actie-ruimte te tekenen:
- $m = 2, n = 3$ is winnend voor de beginnende speler,
- $m = 2, n = 4$ is winnend voor diens tegenstander
- $m = 2, n = 5$ is winnend voor de beginnende speler

Bonus: (5 punten)
Toon aan: - het geval $m = 2, n$ is oneven is winnend voor de beginnende speler
- het geval $m = 2, n$ is even is winnend voor diens tegenstander

Opgave 2. (24 punten)
Gegeven een binaire boom met ingang (ofwel: wortel) `wortel`. Hierin is `wortel` een pointer naar een `knoop`, die er als volgt uitziet:

```
class knoop {
  public:
    knoop* links;
    knoop* rechts;
    knoop* papa;
    knoop* rbroer;
    int jaar;
}; // knoop
```

Voorbeeld:



Bij aanvang van deze opgave zijn de `papa`-velden en de de `rechterbroer`-velden NULL. Het `jaar`-veld van een knoop geeft het geboortejaar van de knoop aan. Deze `jaar`-velden zijn reeds correct gevuld. Het voorbeeld toont alleen de `jaar`-velden.

a. (8 punten)
We noemen twee kinderen van dezelfde ouder een *tweeling* als ze in hetzelfde jaar geboren zijn. Schrijf een *recursieve* C++-functie `int tweeling(knoop* wortel)` die het aantal tweelingen telt. Voor de voorbeeldboom zijn dat er twee.
b. (8 punten)
Schrijf een *recursieve* C++-functie `void gezin(knoop* wortel)` die de `papa`-pointer van elke knoop naar de ouder van die knoop laat wijzen (NULL voor de wortel) en de `rbroer`-pointer naar zijn rechterbroer (NULL als een knoop geen rechterbroer heeft). Merk op dat linkerkinderen een broer kunnen hebben, maar rechterkinderen nooit.
c. (8 punten)
Laten de `papa`-velden en de `rbroer`-velden van de boom met ingang `wortel` correct gevuld zijn. Zij verder gegeven dat de boom *compleet* is, dus alle niveaus op het onderste na zijn helemaal gevuld, en op het onderste niveau zitten alle knopen zo ver mogelijk naar links aangeschoven. We nemen aan dat de boom ten minste twee niveaus heeft. Schrijf nu een *niet-recursieve* C++-functie `void laagste(knoop* wortel)` die het `jaar`-veld van de knopen op het onderste niveau van links naar rechts afdrukt. Loop eerst naar de meest linker knoop op het onderste niveau en gebruik dan de `rbroer` pointers en de `papa`-pointers.

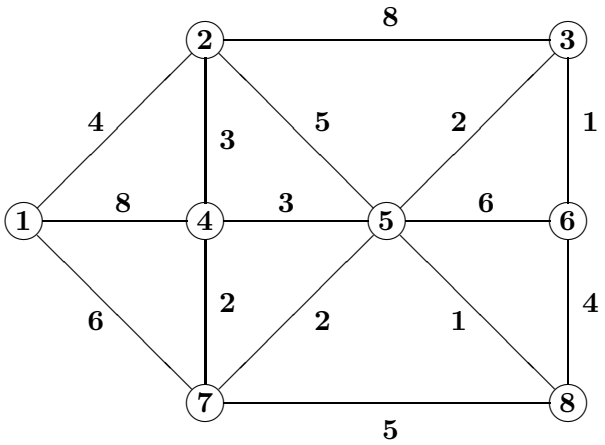
-vervolg op pagina 3

Opgave 3. (18 punten)
Deze opgave gaat over verdeel en heers. De onderdelen **a** en **b** staan los van elkaar.

a. (8 punten)
Gegeven een array $A = A[0], A[1], \ldots, A[n-1]$ dat n karakters bevat. We willen weten of A een palindroom is, d.w.z. of A van links naar rechts hetzelfde is als A van rechts naar links. Voorbeeld: a, b, a, c, b, b, c, a, b, a is een palindroom. Neem aan dat n even is en $n \geq 2$.
Geef een decrease-by-two algoritme dat nagaat of A een palindroom voorstelt. Schrijf daartoe een *recursieve* C++-functie `bool palindroom(char A[ ], int i, int j)`, die het probleem oplost voor het (deel)array $A[i], A[i+1], \ldots, A[j-1], A[j]$ $(j > i)$.
De aanroep `return palindroom(`$A, 0, n-1$`);` geeft dan het antwoord voor het hele array.
b. (10 punten)
Gegeven is nu een array $A = A[0], A[1], \ldots, A[n-1]$ dat gehele getallen bevat. Hierin is $n \geq 2$ een 2-macht. We noemen een index i een *top* als geldt: $A[i-1] < A[i]$ en $A[i+1] < A[i]$. In het bijzonder is een randpunt (begin of eind van het array) dus nooit een top. Met behulp van divide-and-conquer willen we bepalen hoeveel toppen het array A heeft. Het array dient hiervoor in twee gelijke delen te worden verdeeld.
Schrijf een *recursieve* C++-functie `int toppen(int A[ ], int links, int rechts)` die het probleem oplost voor het deelarray A[`links`], …, A[`rechts`] ter lengte een 2-macht.
Voorbeeld: $12, 34, 29, 45, 57, 42, 18, 22$ heeft twee toppen, namelijk op posities 1 (waarde 34) en 4 (waarde 57). Positie 7 is een randpunt en wordt dus niet als top geteld.

Opgave 4. (12 punten)
Het algoritme van Dijkstra bepaalt voor gewogen grafen de (lengtes van) kortste paden vanuit een gegeven knoop naar alle andere knopen.

(i) Pas het algoritme van Dijkstra toe op onderstaande graaf, beginnend in knoop 1. Geef voor elke stap van het algoritme duidelijk aan welke knoop erbij wordt gekozen in U (= verzameling knopen waarvan de kortste afstand vanaf knoop 1 bekend is) en welke labels door die keuze veranderen en hoe. Leg ook uit hoe je uitwerking gelezen moet worden (legenda).
(ii) Geef de uiteindelijke boom van kortste paden met daarin de bijbehorende lengtes van de kortste paden vanaf knoop 1.

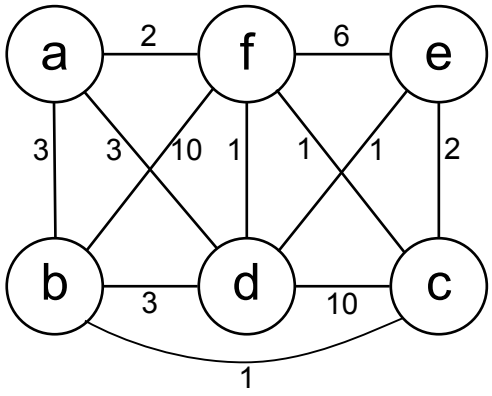


-vervolg op pagina 4

Opgave 5. (26 punten)
a. (10 punten)
Leg uit hoe best-fit-first branch and bound werkt voor minimalisatieproblemen in het algemeen. Geef daarbij o.a. aan hoe (deel)oplossingen gegenereerd worden, wat met branch bedoeld wordt en wat met bound, wat best-fit-first betekent, wanneer gesnoeid wordt, enz.

We gaan nu branch and bound toepassen voor het vinden van een Hamiltonkring met minimaal totaalgewicht in een (niet noodzakelijk volledige) ongerichte graaf met gewichten op de takken. Die gewichten stellen de afstanden tussen de knopen voor. Je mag ervan uitgaan dat de graaf ten minste één Hamiltonkring heeft.

Voorbeeld:



Met afstandstabel:

-	3	-	3	-	2
3	-	1	3	-	10
-	1	-	10	2	1
3	3	10	-	1	1
-	-	2	1	-	6
2	10	1	1	6	-

Uitgaande van een deeloplossing is een ondergrens voor 2 keer de lengte (!) van een Hamiltonkring voor elke (complete) uitbreiding daarvan te berekenen door voor elke knoop de lengtes van de twee kortste takken tussen die knoop en een buurknoop te nemen, en deze bij elkaar op te tellen. Indien een tak (u, v) al in de deeloplossing aanwezig is moet deze in de afschatting bij u en v ook gekozen worden.

b. (4 punten)
(i) Zonder beperking der algemeenheid hoeven we alleen maar Hamiltonkringen te beschouwen waarin knoop b voor knoop c komt. Leg uit waarom dit zo is.
(ii) Geef aan hoe je je oplossingen (Hamiltonkringen dus) genereert en wanneer je een knoop (= deeloplossing) voor dit concrete probleem snoeit.
c. (12 punten)
Pas de methode toe op het voorbeeld en teken de bijbehorende state space tree. Gebruik de gegeven ondergrens en de observatie uit **b**(i). Geef in de state space tree aan in welke volgorde de knopen bekeken worden en welke deeloplossingen gesnoeid worden. Geef bij elke gesnoeide knoop aan waarom deze gesnoeid is. Geef tevens van ten minste vijf deeloplossingen de berekening van de ondergrens.